**深 圳 大 学 实 验 报 告**

**课程名称：­ 算法设计与分析**

**实验项目名称： 动态规划（金罐游戏问题）**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 杨烜**

**报告人： 谢弘烨 学号： 2020151036**

**实验时间： 2022年4月30日 至 2022年5月9日**

**实验报告提交时间： 2022年5月9日**

**教务部制**

|  |  |  |
| --- | --- | --- |
| **一、实验目的：**   1. 掌握动态规划算法设计思想。 2. 掌握金罐游戏问题的动态规划解法。 | | |
| **二、实验原理：** | | |
| **三、实验用品：**  Viusal Studio 2022  MicroSoft Office2019 | | |
| **四、实验过程及内容：**   1. 给出解决问题的动态规划方程； 2. 随机产生金罐的个数和金币值，对小数据模型利用蛮力法测试算法的正确性，并记录A选择的是哪个金罐； 3. 随机产生金罐的个数和金币值，对不同数据规模（n的值）测试算法效率，并与理论效率进行比对，请提供能处理的数据最大规模，注意要在有限时间内处理完； 4. 该算法是否有效率提高的空间？包括空间效率和时间效率。 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **五、实验现象及数据处理：**   1. **问题描述：**   金罐游戏中有两个玩家，A和B，所有的金罐排成一排，每个罐子里都有一些金币， 玩家可以看到每个金罐中有多少硬币。A和B两个玩家交替轮流打开金罐，但是必须从一排的某一端开始挑选，玩家可以从一排罐的任一端挑选一个罐打开。 获胜者是最后拥有更多硬币的玩家。 我们是A玩家，问如何才能使A 收集的硬币数量最大。  假设 B 也是按照“最佳”策略玩，并且 A 开始游戏。   1. **基本思路：**   将罐子中的金子用数组保存，每一次的选择都仅能选择第一个或最后一个罐子，将剩下的罐子作为新的数组交给对手选择。  对于A的每一次选择，由于AB采用相同策略进行游戏，B在A选择后剩下的罐子中的选择是固定的，即对于A的每一次选择，在交给对手选择后还给自己的数组是固定的。  假设对于剩下的数组A已经有方法从中获胜，即对于A选择后剩下的子问题具备最优子结构，  那么令  *表示在下标从0到n的罐子中能选到的最大金子数*  *表示第i个罐子里的金子数，且i每次仅有数组首尾两元素的下标可选*  则：  此处以及表示A选择后经过B选择再次交给A的罐子中可获得的最大金子数  以变量，替代上述公式中的0，n，可以得到原问题的动态规划方程，  即：  当，相遇时，即时，代表当前只有一个罐子，那么就直接选择这个罐子，则：  综上：  伪代码：  //从对角线开始，一行一行向右上角填表  for(i = 0; i < n; i++)  for(l = 0; l < n; l++)  left = l  right = l + iss  if(I = 0)  g[left, right] = value[left]  else  g[left, right] = max(v[left] + g[left + 1, right],  v[right] + g[left, right -1]  时间复杂度为O(n2)  空间复杂度为O(n2)  以上述思想为基础编写程序，对材料中的两组例子有：    对于随机生成的金罐序列，利用蛮力法验证：    其中，蛮力法的伪代码实现：  PROCESSBYFORCE(left, right):  //只要元素可选就选，返回sum的最大值  if(canChoose(A, left))  //如果第一个元素可选就选第一个元素  disable(left)  sum+=v[left]  if(canChoose(B, left + 1)  disable(left+1)  PROCESSBYFORCE(left + 2, right)  recover(left + 1)  else  disable(right)  PROCESSBYFORCE(left + 1, right – 1)  recover(right)  recover(left)  sum-=v[left]  else  //与上述情况对称，只要最后一个元素可选就选择最后一个元素  由于该蛮力法不包含任何策略，在小数据情况下可以得出正确答案，数据规模增大后不能保证正确性。  当n=4时进行10000000次蛮力法验证，结果依旧相同。可见算法基本正确  效率分析：  当n为103至104时：   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | 1000 | 2000 | 3000 | 4000 | 5000 | | 理论 | 17.77 | 71.06 | 159.89 | 284.24 | 444.13 | | 实测 | 14.85 | 75.60 | 164.35 | 278.50 | 436.25 |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 6000 | 7000 | 8000 | 9000 | 10000 | | 639.55 | 870.50 | 1136.98 | 1438.99 | 1776.53 | | 639.55 | 877.10 | 1154.45 | 1477.85 | 1843.80 |   实测值与理论数据贴合较好。然而由于其过高的空间复杂度，不仅使得该过程的时间复杂度因为硬件限制导致表现不够理想，更使其可处理的最大规模数据仅达到50000。因此还需要对算法进行进一步优化。   1. **优化思路：**   由于该算法执行过程实际是对一个n\*n规模的状态表进行填写的过程，而过高的空间复杂度使得计算机在执行算法的过程中耗费大量时间在读写数据中，进而导致其效率表现不佳。  在填表过程中可以发现，在对每一条斜边的处理过程中，我们仅需要其前面两行斜边的数据，其余元素可以省去。  如：   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 4 | 6 | 2 | 3 | | 4 | V0  4 | V1  6 |  |  | | 6 |  | V1  6 | V1  6 |  | | 2 |  |  | V2  2 | V3  3 | | 3 |  |  |  | V3  3 |   在填写绿色元素的过程中，我们仅需要读取黄色元素中记录的玩家处理本元素代表的罐子时的选择，以及读取红色元素中玩家在本元素代表的罐子中能拿到的最大金子数。  综上所述，该算法过程中状态表的规模可以缩减为3\*n，即该算法空间复杂度优化为O(n)。然而，由于该优化过程仅缩减状态表的规模，算法执行过程中仍需遍历所有的状态，所以优化过后的时间复杂度仍为O(n2)。  由于该优化方法仅修改访问状态表的方式，算法核心部分与前文相同，故不展示伪代码。  效率分析：  当n为103至104时：   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | 1000 | 2000 | 3000 | 4000 | 5000 | | 理论 | 14.71 | 58.83 | 132.36 | 235.30 | 367.66 | | 实测 | 14.25 | 59.30 | 138.05 | 244.60 | 377.40 |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 6000 | 7000 | 8000 | 9000 | 10000 | | 529.43 | 720.62 | 941.22 | 1191.23 | 1470.65 | | 551.95 | 726.40 | 929.25 | 1175.55 | 1470.65 |   可见实测效率与理论值贴合程度较好。由于该优化过程未改变算法核心代码，效率曲线趋势仍旧呈现n2形式。此外，优化过后算法可处理最大规模达到78000。   1. **效率比较**   当n为103至104时：   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | 1000 | 2000 | 3000 | 4000 | 5000 | | 优化前 | 14.85 | 75.60 | 164.35 | 278.50 | 436.25 | | 优化后 | 14.25 | 59.30 | 138.05 | 244.60 | 377.40 |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 6000 | 7000 | 8000 | 9000 | 10000 | | 639.55 | 877.10 | 1154.45 | 1477.85 | 1843.80 | | 551.95 | 726.40 | 929.25 | 1175.55 | 1470.65 |   可见，尽管优化后时间复杂度仍为O(n2)，实际运行时间也有所减少。可能原因为：随着空间复杂度减小，算法执行过程可以在内存的高速缓存区中执行，避免了不断读写数据的资源消耗。 |
| **六、实验结论：**  此次实验主要考察对动态规划思想的掌握。在利用动态规划解决问题时，应该先确定原问题与子问题的关系，找到终止条件，进而确定原问题的动态规划方程，最后依照动态规划方程进行编程具体实现。  此次实验中还找到了一个对算法实际运行效率进行优化的新方向：对算法的空间复杂度进行优化。通常情况下想要提高算法的运行效率就要优化其时间复杂度，然而有时算法的时间复杂度已经无法进一步优化，而其实际运行效率仍不理想，这可能是因为算法过高的空间复杂度导致了在硬件层执行算法的过程中消耗大量资源。此时优化算法的空间复杂度就可以使硬件在执行算法的过程中减少不必要资源的消耗，充分利用高性能硬件，进而提高算法的时间运行效率。 |

|  |
| --- |
| **思考题：** |
| **指导教师批阅意见：**  **成绩评定：**  **指导教师签字：**  **年 月 日** |
| **备注：** |

**注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。**

**2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内**。